(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number : **0 620 517 A1**

(12) EUROPEAN PATENT APPLICATION

(21) Application number : **94302629.4**

(22) Date of filing : **13.04.94**

(51) Int. Cl.[5] : **G06F 3/033**

(30) Priority : **15.04.93 US 48445**

(43) Date of publication of application :
19.10.94 Bulletin 94/42

(84) Designated Contracting States :
DE FR GB

(71) Applicant : **International Business Machines Corporation**
Old Orchard Road
Armonk, N.Y. 10504 (US)

(72) Inventor : **Morgan, Scott Anthony**
3609 Kentfield Road
Austin, Texas 78759 (US)
Inventor : **Stone, Steve Stanley**
2303 Dowd Lane
Austin, Texas 78728-6715 (US)
Inventor : **Swearingen, Craig Ardner**
8905 Martha's Drive
Austin, Texas 78717 (US)

(74) Representative : **Moss, Robert Douglas**
IBM United Kingdom Limited
Intellectual Property Department
Hursley Park
Winchester Hampshire SO21 2JN (GB)

(54) Object resizing and repositioning for a new font in a graphical user interface.

(57) A graphical user interface is changed according to a new font for one object in the graphical user interface. The set of appearance parameters are adjusted for the object according to the new font and presenting the object according to the adjusted appearance parameters and the new font. A set of related objects within the interface can also be changed according to the new font. Once the new font is established for the object, the set of related objects in the graphical user interface for the object is determined. Then, each respective set of appearance parameters for the set of related objects is adjusted according to its object class and the set of related objects is presented according to their respective appearance parameters and the new font.

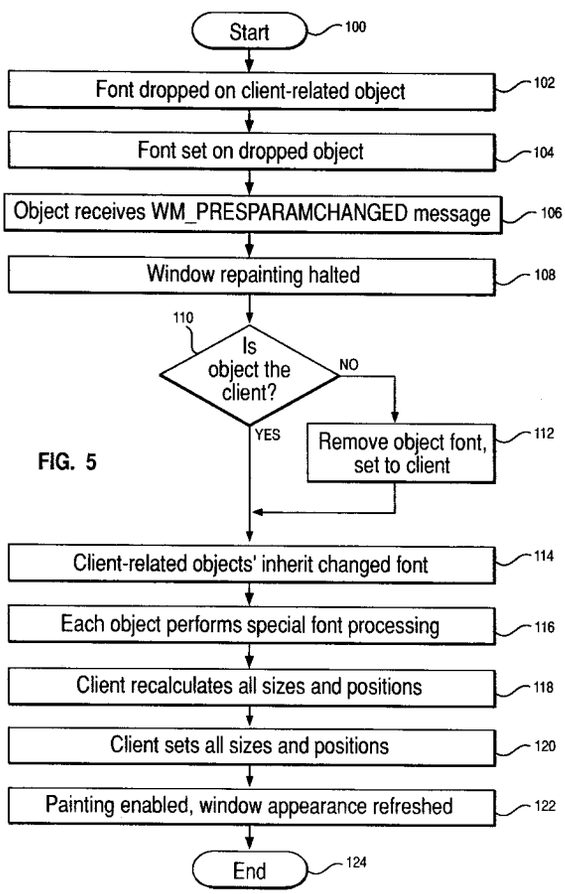


FIG. 5

EP 0 620 517 A1

This invention relates generally to a graphical user interface to control a computer system. More particularly, it relates to changing appropriate objects within the graphical user interface in response to a request to change a font on one of the objects.

It is well known to provide a graphical user interface to allow a user to control a computer system and to present the results of user actions on the system display. In a graphical user interface, applications and data files are generally presented within windows. Each window typically has a number of elements which are associated with the window and are presented in the display space allocated to the window. The window and each of its separate elements are generally structured and stored in memory as objects which are related logically. The elements of the window are generally child objects of the main window object (for example the individual screen elements representing specific processes created solely for a specific 'parent' process).

In newer graphical user interfaces, it is possible for a user to select the font by which the information is displayed if the standard font is not pleasing to the user. Typically, after the selection of the new font, the window is refreshed to display the new font, but is not changed in any other way. If the font was a larger font, it is often clipped or unreadable due to the element within the graphical user interface being too small to display it properly. If a smaller font was selected, then the window display space is wasted. Furthermore, the GUI looks awkward and malproportioned. Further, when a font is selected for one object in the graphical user interface, the font is typically changed only for the immediate object. When that object is a child object of the main window or another parent object within the window, sibling objects of the object receiving the font will not receive the new font even when it makes sense that they should.

The present invention provides a system and a method for changing a graphical user interface according to a user selection of a new display font, the system comprising:

means responsive to a user input for changing a font for a first object of a first object class displayed in the graphical user interface to a new font;

means for automatically adjusting a set of size and position parameters for the first object according to the new font and the first object class; and

means for presenting the first object according to the adjusted size and position parameters and the new font.

A method according to the invention comprises changing a font for an object in the graphical user interface to a new font, adjusting a set of appearance parameters for the object according to the new font and presenting the object according to the adjusted appearance parameters and the new font. The appearance parameters of the object are adjusted according to an object class of the object. A font palette within the graphical user interface may be used to change the font, wherein the new font is 'grabbed' and 'dragged' from a font palette and 'dropped' over the object. A font is 'grabbed' by selecting, using a cursor and positive selection, an icon or name representative of the font from a system menu. 'Dragging' is the movement of such a selected item, such as by moving a mouse after the selection associates the mouse cursor with the item. 'Dropping' is using the cursor positioning to indicate what object the new font is to be associated with.

It is preferred that the new font is passed to objects which are logically related in the graphical user interface to the object for which the new font was selected. A set of related objects within the interface can preferably also be changed in size and position according to the new font. Once the new font is established for the object, the set of related objects in the graphical user interface for the object is determined. Then, each respective set of appearance parameters for the set of related objects is adjusted according to its object class and the set of related objects is presented according to their respective appearance parameters and the new font. The set of related objects can be determined by determining a set of child objects of a parent object for the object, the set of child objects being the set of related objects.

Specific Description

The present invention will now be described in more detail, by way of example, with reference to the accompanying drawings in which:

FIG. 1 shows a computer comprising system unit, keyboard, mouse and display.

FIG. 2 is a block diagram of the components of the computer shown in FIG. 1.

FIGS. 3A and 3B depict a prior art graphical user interface in which the controls of a window are not resized in response to the selection of a larger font.

FIGS. 4A and 4B show the graphical user interface of the invention in which the appropriate objects within an interface are resized and repositioned according to the selection of a larger font.

FIG. 5 is a flow diagram of dropping a new font on a window and adjusting that portion of the interface.

FIG. 6 is a flow diagram of adjusting other child objects within the window.

FIG. 7 is a diagram of related objects within a window.

FIG. 8 is a flow diagram of adjusting child objects of other windows in an application.

The invention may be run on a variety of computers or collection of computers under a number of different operating systems. The computer could be, for example, a personal computer, a mini computer,

mainframe computer or a computer running in a distributed network of other computers. Although the specific choice of computer is often limited only by disk and disk storage requirements, computers in the IBM PS/2 (TM) series of computers in particular could be used in the present invention. For additional information on IBM's PS/2 series of computers, the reader is referred to Technical Reference Manual Personal Systems/2 Model 50, 60 Systems IBM Corporation, Part No. 68X2224 Order Number S68X-2224 and Technical Reference Manual Personal Systems/2 (Model 80) IBM Corporation Part No. 68X 2256 Order Number S68X-2254. One operating system which an IBM PS/2 personal computer may run is IBM's OS/2 2.0 (TM) for more information on the IBM OS/2 2.0 Operating System the reader is referred to OS/2 2.0 Technical Library, Programming Guide Vol. 1, 2, 3 Version 2.00 Order Nos. 10G6261, 10G6495, 10G6494.

In the alternative, the computer system might be in the IBM RISC System/6000 (TM) line of computers which run on the AIX (TM) operating system. The various models of the RISC System/6000 is described in many publications of the IBM Corporation for example, RISC System/6000, 7073 and 7016 POWERstation and POWERserver Hardware Technical reference, Order No. SA23-2644-00. The AIX operating system is described in General Concepts and Procedure--AIX Version 3 for RISC System/6000 Order No. SC23-2202-00 as well as other publications of the IBM Corporation.

In FIG. 1, a computer 10, comprising a system unit 11, a keyboard 12, a mouse 13 and a display 14 are depicted. The screen 16 of display device 14 is used to present the visual changes to the data object. The graphical user interface supported by the operating system allows the user to use a point and shoot method of input by moving the pointer to an icon representing a data object at a particular location on the screen 16 and press one of the mouse buttons to perform a user command or selection.

FIG. 2 shows a block diagram of the components of the personal computer shown in FIG. 1. The system unit 11 includes a system bus or plurality of system buses 21 to which various components are coupled and by which communication between the various components is accomplished. The microprocessor 22 is connected to the system bus 21 and is supported by read only memory (ROM) 23 and random access memory (RAM) 24 also connected to system bus 21. A microprocessor in the IBM multimedia PS/2 series of computers is one of the Intel family of microprocessors including the 386 or 486 microprocessors. However, other microprocessors including, but not limited to, Motorola's family of microprocessors such as the 68000, 68020 or the 68030 microprocessors and various Reduced Instruction Set Computer (RISC) microprocessors manufactured by IBM, Hewlett Packard, Sun, Intel, Motorola and others may be used in the specific computer.

The ROM 23 contains among other code the Basic Input-Output system (BIOS) which controls basic hardware operations such as the interaction and the disk drives and the keyboard. The RAM 24 is the main memory into which the operating system and application programs are loaded. The memory management chip 25 is connected to the system bus 21 and controls direct memory access operations including, passing data between the RAM 24 and hard disk drive 26 and floppy disk drive 27. The CD ROM 32 also coupled to the system bus 21 is used to store a large amount of data, e.g., a multimedia program or presentation.

Also connected to this system bus 21 are various I/O controllers: The keyboard controller 28, the mouse controller 29, the video controller 30, and the audio controller 31. As might be expected, the keyboard controller 28 provides the hardware interface for the keyboard 12, the mouse controller 29 provides the hardware interface for mouse 13, the video controller 30 is the hardware interface for the display 14, and the audio controller 31 is the hardware interface for the speakers 15a and 15b. Also coupled to the system bus 21 is digital signal processor 33 which corrects the sound produced by the speaker system and is preferably incorporated into the audio controller 31. The speakers 15a and 15b may be used to present audio objects to the user. An I/O controller 40 such as a Token Ring Adapter enables communication over a network 46 to other similarly configured data processing systems.

One of the preferred implementations of the present invention is as a set of instructions in a code module resident in the random access memory. Until required by the computer system, a set of instructions may be stored on a computer readable medium, for example, the hard disk in hard disk drive 26, optical disk in the CD ROM 32 or a floppy disk in the floppy disk drive 27. As shown in the figure, the operating system 50 and a software component responsible for the provision of a graphical user interface (here shown as Presentation Manager ($_{TM}$) 52) are resident in RAM 24. In this example, the invention is embodied in a font handler 54 which is an adjunct onto the operating system or Presentation Manager software component. Presentation Manager is a trademark for IBM's graphical user interface software, embedded in the OS/2 operating system. Some other systems use graphical user interface providing software which is a stand alone piece of code. If the present invention is implemented in a stand alone application which has its own graphical user interface, a similar code module would have to be developed for that particular application.

The problems of the prior art graphical user interfaces are illustrated in FIGS. 3A and 3B. An initial win-

dow 60 is shown before any font changes have taken place. As shown, the font and subobjects within the window are appropriately sized and positioned and the graphical user interface has a pleasing appearance. Nonetheless, the user has decided that he desires a larger font. In FIG. 3B, the window 62 is shown after a larger font has been dropped on client area 64 (The body of the window inside the border and below the action bar is the client area - it is the workspace for viewing, entering, and selecting information). As shown, the font is too large for entry fields 66 and buttons 68. Buttons are screen elements displaying a word or phrase that indicates a function that will be performed if the button is selected (e.g. by mouse cursor position and mouse button press). The text information in these control windows (hereinafter controls) is clipped and information is lost. Further, the appearance of the graphical user interface is not pleasing and does not impart the needed information to the user. Notice that the font is limited to the client area and its children and is not extended to other text containing objects such as the information line 69.

The example shown in FIGS. 3A and 3B assumes the font was dropped on the client area rather than on a particular control. Dropping on the client, all of the client's children will also inherit the new font. If dropped on a control, only that control would get the new font, plus, of course, any children of the control. In standard operation of OS/2, fonts are a presentation parameter. Presentation parameters are inherited by children unless the child has had its presentation parameter already explicitly set.

The graphical user interface of the present invention is depicted in FIGS. 4A and 4B. The window 70 in the graphical user interface depicted in FIG. 4A is similar to that in FIG. 3A. A new font is being dragged from the font palette 71 and is about to be dropped on the window 70 by pointer 73. The pointer 73 resembles a pencil indicating that a new font is being dropped. In FIG. 4B, the new font which is larger than the prior font, has been dropped on the client area 74 of the window 70. The window 70 and the client area 74, the entry fields 76 and buttons 77 have all been resized to accommodate the larger font. Further, the entry fields 76 and text 78 have been repositioned within the larger client area 74.

In the invention, the font is flowed to all related areas regardless of whether they are Presentation Manager children or not. As an example, the frame lines are not children of the client area, but they get the new font because they are inherently related to the client area. A font dropped anywhere within the client area of a window with font support of the invention, whether on the client area or on any control within the client, will flow the font to all controls in the client and the client itself. And, or course, the font processing will cause the client and all the controls to resize and reposition to best display the new font.

The information line is just another text-based adjunct to the window. The user sees it as part of the whole. Its text changes as the user moves the cursor around the client area from one control to another. Though in the Presentation Manager parent-child hierarchy it is not a child of the client, it is considered to be related to the client in the preferred embodiment of the invention, and so it is treated as such by changing its font to match that of the client.

Note, however, that the text within the title bar 80 and action bar 82 have not been resized as these objects are not logically related to those within the client area 74.

The only changes made to the title bar and action bar as a result of a font change to the client is to change their width due to a change in the width of the frame window. This size change is not a result of the font change, but rather solely as a result of the frame size changing. The same thing happens when the user resizes the frame with the mouse.

The OS/2 Font Palette 71 is a tool which is part of the OS/2 2.0 operating system that is installed in the System Setup folder. The System Setup folder is found in the OS/2 System folder which is an object on the Workplace Shell desktop. This tool is used to change the font an object is using. The tool shows a series of system declared fonts from which to choose. If the font the user is looking for is not found within this series, the user can press the Edit push button which displays a dialogue requesting information about the font in which the user is interested.

Once the desired font is available, the user can, using the drag mouse button, drag a font and drop it on the object that should use that font. When this occurs, the Font Palette determines what object the mouse is over and retrieves its Presentation Manager window handle. With this window handle, the Presentation Manager WinSetPresParam API is used with the PP_FONTNAMESIZE option specifying the font string associated with the one desired. The font string is in the format pointsize.facename.emphasis which is what is displayed in the list provided by the Font Palette.

Not only does the Font Palette support editing the font shown but it also allows you to add additional fonts to the system through this manner. The Edit font dialogue provides an option for adding a font to the list of known system fonts. This request prompts another dialogue which requests the file name where the font definitions exist.

While the Font Palette is the means used in the OS/2 2.0 Workplace Shell for the user to select a new font and indicate the window to which that font should apply, its use is not required to enable the font changes described in this application. Those skilled in the art could conceive of several ways of commanding the operating system to change a font in its GUI. In OS/2, the Presentation Manager notification of the

new font presentation parameter is done by the WM_PRESPARAMCHANGED message, caused by the issuance of the WinSetPresParam Presentation Manager API.

The process by which a window is adjusted according to a new font is depicted in the flow diagram of FIG. 5. The process begins as the font is being dragged from the font palette and dropped on the window, steps 100 and 102. When this occurs, the operating system determines what object the font was dropped on and sets the font of the object to the new font in step 104. Next, the object on which the font was dropped receives a WM_PRESPARAMCHANGED message notifying it that its font has changed, step 106. This message is the OS/2 message, other operating systems and applications would have other syntax to notify the object that its font had changed. At this point in the process, the window repainting is halted in step 108. Painting is halted so that the user does not see each control change its font, resize and position itself as required, one after the other. This would make the window flash and look confused until all the controls had finished their processing and all was sorted back out in the final arrangement. Painting is enabled again once all of these calculations and changes are finished, then refresh the display so that the user sees only one repaint into the final, new font appearance.

In step 110, a test is performed to determine whether the object is the parent object within in a tree of related objects. One example, is the client area which is the parent object of all the controls and text fields which appear within the client area. If the object is not such a parent object, the object in step 112 removes the font from itself so it can inherit the font changes from the parent object and sets the parent object to the new font.

If the object is the parent object of the particular group of related objects, in step 114, all the child objects, e.g., controls, inherit the new font and are notified by the operating system that their font has changed. Each control performs any special processing which it requires to perform a font change, step 116. The parent object recalculates its own size and position and all control size and position within the main window based on the new font characteristics, step 118. In step 120, the client area sets all sizes and positions as calculated. Finally, in step 122, window repainting is reenabled and the window is refreshed to show the new sizes, position and font. The process ends in step 124. The process by which child objects of the parent object in the GUI adjust their appearance in response to the parent object being set to a new font is depicted in FIG. 6 as steps 114 and 116. The process begins in step 130 as the parent object's font is changed. In step 132, the parent object is notified with a WM_PRESPARAMCHANGED message. Step 134 begins the loop in which each of the information lines, status lines and security lines are examined. A test is performed in step 136 to determine whether the font is different from the client areas object's font. If so, the object is set to the new font in step 138. If not, a second step is performed to determine whether the new font is different from the current window font, step 140. Step 136 relates only to the information, status and security lines. Step 140 relates only to the client area itself. In step 140, the client area checks to see if the new font for the client area is the same or different from the current font for the client area. If the new font and the current window font are the same, in step 142 the message is thrown away and the processing ends. If they are different, the font change message is passed on to the child object, step 144. All controls in the body of the child object inherit the new font from the parent object and receive the font change message in step 146. Finally, in step 148, each child object performs this specific processing for the new font and passes a font change message onto the operating system. The process ends, step 150.

As discussed above, each of the child object's processing may differ depending on the class of the object. Also, other objects in the GUI, which are not adjusted because of the new font as they contain no text or are not deemed to be related to the object on which the font was dropped, must be adjusted in size because of the objects which have changed, the related objects, due to the new font.

When a font change occurs, the affected objects should resize and adjust other aspects as needed, to reflect the new font to its best advantage. Certain classes of controls need unique adjustments for a new font size, as follows:

- Entry fields should adjust their height and width to accommodate the new font.
- List boxes should adjust the height of each list item, adjust the range and position of its scroll bars, and scroll the list items if needed.
- Static text fields, radio buttons and check boxes should grow or shrink both vertically and horizontally so that the needed space to display the text in the new font is used.
- The height of any information line, status line and/or security line should be adjusted.
- The height of the scroll arrows on scrollable output fields and entry fields with scroll arrows should be adjusted to reflect the new height of the related control.
- The entry field and list portions of combo boxes should adjust in the same manner as stand alone controls of those types.
- Notebooks should adjust the width and height of their tabs so all of the tab text remains visible.
- Containers should reflow their current view to reflect the new font.

- Progress indicators should resize and respace their tick marks so that any tick text is still properly displayed.

The changes performed by each object are defined with font-based parameters. Each object knows what to do given a "general" font change because when created, the object was made size font specific. For example, an entry field is defined by how many average character widths it should contain; a list box by how many lines of max character height it should display. So, each control knows inherently what its size is based on the current font. When a "particular" font change occurs, it recalculates what its size should now be based on the new font characteristics.

Once all of the controls have resized appropriately for the new font, they should be reflowed within the window itself so that they are still properly spaced in relation to each other. Non-sizeable windows should also be resized to reflect this new flow of the controls. Sizeable windows need to adjust the size and position of their scroll bar sliders to reflect the new size of the scrollable region.

By resizing controls affected by a font change to their optimum size for that font, then reflowing them within their window to retain optimum spatial alignment, the window after the font change will retain its readability, usability and balanced appearance.

A panel's layout is described using tags in a panel definition file. In Presentation Manager, a panel is defined by specifying at what exact coordinates each control is placed, and exactly how large it is in pels. In the panel definition files, the panel is defined in terms of general panel layout, such as columns and rows, expandability, control relationships, related text such as prompt, prefix and suffix, and the like. Each control's size is defined in character/font based terms as described above.

With this information, the panel is dynamically laid out at run time based on the general column/row/relationship layout, and based on the current font and the length of the current text strings in use for text controls (static text, output text, prompt text, check box and radio button text...). The actual text would change in different languages, for example. Once these calculations are made, the controls are set in terms that Presentation Manager can understand, i.e., located at exact coordinates and sized in pels.

When a font change occurs, the window is redisplayed based on the column/row/relationship definitions and the new font characteristics.

The dynamic creation of a graphical user interface based on new parameters given to an operating system at run time is discussed in the copending, commonly assigned application filed August 19, 1991, Serial No. 07/747,167, entitled "Computer System for Dynamically Generating Display Screen Panels Providing Interactive Interfaces for Application Program Steps", to J. W. Malcolm, which is hereby incorporated by reference.

Each font and each window have certain parameters which are used to determine the best graphical user interface for the combination of the font and the window. A Font Parameters Table and Windows Parameters Table are given below as an illustration of the types of parameters which a graphical user interface might use.

Font Parameters Table

AvCharW
MaxCharW
HUpperChar
HCharBox
TableWChar
Prop

The font parameters table includes parameters for a particular font for the average character width, the maximum character width, the height of the uppercase characters, the height of the character box, a table of the width of each character in the alphabet for the font and whether the font is proportional or nonproportional. All of these parameters will determine the optimum size for the affected controls for objects within the graphical user interface.

Window Parameters Table

SzWin
PosWin
$SzCon_1$, $SzCon_2$, $SzCon_3$...
$PosCon_1$, $PosCon_2$, $PosCon_3$...

The Window Parameter Table will have values for the size of the window, the position of the window and the graphical user interface and the size and position of each of the controls or subobjects within the window. Each of these parameters may change when a new font is dropped on an object in the window.

The actual objects which are considered "related" is a design choice by the programmer. In the examples above, when the user drops a new font on the client area of a window, it indicates a desire to change the font of those aspects of the panel which share a common factor, e.g., those that relate to data input and output. Therefore, all of the parts of the client area that relate to this should have their font changed, thus keeping similar objects acting and looking like a logical, integrated whole. Parts of the window that do not share this common factor should not be changed unless the user specifically drops a new font on them as well.

In the preferred embodiment of the present invention, the text in the information line, status line and/or security line should also inherit the new font if dropped on the client area, since their role is to output data to the user.

However, in response to the font being dropped on an object in the client area, the action bar should not inherit the new font. The action bar is primarily a traversal technique for menu display, which is different from the uses of text in the body panel. The font of the title bar should not change. The user may desire that title bars stay consistent throughout the system, as it presents a system type datum. The height of the horizontal and width of the vertical scroll bars should not change, since these sizes are not text-related.

If a new font is dropped on any object other than one of those that is considered related and for which special processing is provided, it will only get the default Presentation Manager processing. For the frame, that means if it is dropped on any frame element besides the frame itself, e.g., title bar or action bar, only that object will reflect the new font. If it is dropped on the frame (sizing border) itself, the frame's children will inherit the new font as well (but will not resize and reposition themselves based on the new font). In the explenary architecture, the client area is not a child of the frame and dropping on the frame has no effect on the client area or its objects.

A table of OS/2 functions which are used in process described in FIGs. 5 and 6 follows below. Similar functions in other operating systems could be used. If the process were implemented in the GUI of a stand alone application, some of the functions might need to be created.

Table of OS/2 Functions

GpiQueryFontMetrics	116,118,148
GpiQueryWidthTable	116,118,148
WinCalcFrameRect	118
WinEnableWindowUpdate	108,122
WinInvalidateRect	122
WinIsChild	110
WinIsWindowVisible	108
WinMapDialogPoints	118
WinQueryPresParam	136,140
WinQueryWindow	110
WinQueryWindowPos	116,118,148
WinRemovePresParam	112
WinSendMsg	116,148
WinSetPresParam	112,138
WinSetWindowPos	116,120,148
WinUpdateWindow	122

GpiQueryFontMetrics: returns the specifics about a given font

GpiQueryWidthTable: returns a table of widths for every character for a given font

WinCalcFrameRect: calculates what size a window frame needs to be for a given size client area

WinEnableWindowUpdate: enables and disables window repaints and refreshes

WinInvalidateRect: sets an area as being in need of being repainted

WinIsChild: queries if a given window is a child of another

WinIsWindowVisible: returns if a window allows refreshing/repainting of its appearance

WinMapDlgPoints: translates values from pels to dialogue units

WinQueryWindow: returns the parent or owner of a given window

WinQueryWindowPos: returns the size and position of a given window

WinRemovePresParam: removes a presentation parameter, such as a font, from a given window

WinSendMsg: sends a message with two parameters to a given window

WinSetPresParam: sets a presentation parameter, such as a font, for a given window

WinSetWindowPos: sets the size and position of a given window

WinUpdateWindow: causes a window to repaint any areas that have been marked as needing it

One of the requirements of the present invention is that the related objects are known or can be determined by the system. The relationship of the objects within a generalized window is shown in FIG. 7. The main window, 200, has three child windows; the client area 202, the frame control 204 and the frame line object 206. The client area further has a plurality of child objects including entry fields 208, text fields 210 and push buttons 212. The frame control object 204 has title bar 214 and action bar 216 as child objects, and the frame line object 206 has information line 218, status line 220 and security line 222 as child objects. In an operating system, such as OS/2, this sort of information is kept by the operating system. Thus, it is relatively easy when a font is dropped on, for example, an entry field 208, to go to the client area 202 to determine all of the related child elements of the client area 202.

However, the structure provided by the operating system may not, in all cases, have the necessary structure to carry out the present invention. For example, there may be certain objects within the client area which are not considered text elements or related to other elements within the client area. A statement would have to be added to the code which implements the invention that all objects which are child objects of the client area, except for a certain class of objects, would inherit the font. Similarly, there may be no root or master object which is apparent to all of the related objects in a particular area of the window. For example, the frame line object does not exist in OS/2. Therefore, for the data structure used by the invention, these objects would have to be defined and inserted.

In a stand alone application or where too many variations on the existing data structure of objects in the operating system are required, it may be more convenient to construct a table of the related elements which will be searched when a new font is dropped on an object. For example, the table below corresponds to FIG. 7.

Table

Main Window

- Client Area
  - Entry Field 1
  - Entry Field 2
  - Entry Field 3
  - Text Field 1
  - Text Field 2
  - Text Field 3
  - Push Button 1
  - Push Button 2
  - Push Button 3
- Frame Control
  - Title Bar
  - Action Bar
- Frame Line
  - Information Line
  - Status Line
  - Security Line

Thus, the objects contained in the table can be varied as the programmer deems appropriate. In this case, the process refers to the table for the particular window to learn to change the fonts in all the child objects of a second order object, such as the client area, the frame control or the frame line object.

Alternatively, the table may resemble that below:

Client Area (EF1,EF2,EF3,TF1,TF2,TF3,PB1, PB2, PB3)

Entry Field 1 (Client Area, EF1,EF2,EF3,TF1,TF2, TF3,PB1, PB2,PB3)

Text Field 1 (Client Area, EF1,EF2,EF3,TF1,TF2, TF3,PB1, PB2,PB3)

Status Line (Information Line, Security Line)

In this table, the code searches for the object on which the font was dropped and finds a listing of the related objects in the parentheses following the object. In the table, EF1, for example, would represent Entry Field 1, TF2 would represent Text Field 2; and so forth. Note that in such table the "related" objects do not have to be the same set of objects which the operating system associates as child objects of a particular parent object in the graphical user interface. Instead, the programmer may decide on an intuitive basis what objects are related to a particular aesthetic characteristic. Other tables and procedures for listing

and determining the related objects would occur to one skilled in the art.

The embodiment above describes run time font changes which flow the font only within the frame window on which the font was dropped.

The invention can also be used to flow a run time font change between all or some of the windows of an application, as shown in FIG. 8. The process begins in step 250. All of the top-level frame windows in the system are determined. Top-level frame windows are those windows that are direct children of the desktop. That is, they are not children of any other window. They can be found by enumerating all direct children of the desktop. Unseen windows of an application are still descendants of the top-level frame or are in the top-level frame's ownership chain, and so the top-level frame can handle their font change. For each top-level frame window found, query and compare its process ID to that of the window which is propagating its font change step 252. If they are the same, they are in the same process, and probably in the same application. Thus, in step 254, the system sends a private message that would only be processed by members of the same application, with a pointer to the new font string as one of its parameters. If the windows do not have the same ID, they are not in the same process, and so should not receive the font, step 256.

If the window receiving this private message has a font different from that to which it is being told to change, step 258, it will issue a WinSetPresParam call to itself, setting itself to use the new font, step 260. If it already has the requested font, no action would be taken, step 256.

Those windows which needed to change to the new font would then handle the font change as if the user had dropped a font on them as described above. First, the related child objects would be determined in step 262 and the font would be changed in those objects in step 264. Part of the private message sent in step 254 should include either the class of the object on which the font was dropped or the class of the parent object used to determine the related objects in the first main window. In step 262, the receiving main window will use this information by consulting its own data structures and tables to determine the related child objects.

Claims

1. A method for changing a graphical user interface display according to a user selection of a new display font, comprising the steps of:

   in response to a user input, changing a font for a first object of a first object class in the graphical user interface to a new font;

   automatically adjusting size and position parameters for the first object according to the font change and the first object class; and

   presenting the first object according to the adjusted size and position parameters and the changed font.

2. A method according to claim 1 which further comprises the steps of:

   determining a set of related objects in the graphical user interface for the first object;

   automatically adjusting size and position parameters for each object of the set of related objects; and

   presenting the set of related objects according to their respective size and position parameters and the changed font.

3. A method according to claim 2 wherein the size and position parameters for each object of the set of related objects are adjusted according to the object class of the respective object.

4. A method according to claim 3, wherein object size parameters are made font-specific when an object is created, with reference to a non-font-specific object class definition, for all classes of object which can be affected by font changes, thereby to facilitate adjustment of size parameters when a new font is selected.

5. A method according to any one of claims 2 to 4 wherein the determining step is accomplished by determining a set of child objects of a parent object for the first object, the set of child objects being the set of related objects.

6. A method according to any one of the preceding claims which further comprises the steps of:

   determining a set of windows which belong to a first process which owns the first object; and

   adjusting size and position parameters for corresponding sets of related objects within each of the set of windows.

7. A method according to any one of the preceding claims wherein the step of changing a font comprises the steps of:

   grabbing and dragging a new font from a font palette in the graphical user interface; and

   dropping the new selected font over the object.

8. A system for changing a graphical user interface display following user selection of a new display font, comprising:

   means responsive to a user input for changing a font for a first object of a first object class displayed in the graphical user interface to a new font;

means for automatically adjusting a set of size and position parameters for the first object according to the new font and the first object class; and

means for presenting the first object according to the adjusted size and position parameters and the new font.

9. A system according to claim 8 which further comprises:

means for determining a set of related objects in the graphical user interface for the first object;

means for adjusting each respective set of size and position parameters for each object of the set of related objects; and

means for presenting the set of related objects according to their respective size and position parameters and the new font.

10. A system according to claim 9 wherein the size and position parameters for each object of the set of related objects are adjusted according to the object class of the respective object.

11. A system according to claim 10, having means for storing non-font-specific size definitions for each class of object which can be affected by font changes, to facilitate adjustment when a new font is selected of font-specific object size parameters.

12. A system according to any one of claims 9 to 11, wherein the means for determining includes means for determining a set of child objects of a parent object for the first object, the set of child objects being the set of related objects.

13. A system according to any one of claims 8 to 12 which further comprises:

means for determining a set of windows which belong to a first process which owns the first object; and

means for adjusting size and position parameters for corresponding sets of related objects within each of the set of windows.

14. A system according to any one of claims 8 to 13 wherein the means for changing a font comprises:

means for grabbing and dragging a new font from a font palette in the graphical user interface; and

means for dropping the new selected font over the first object.

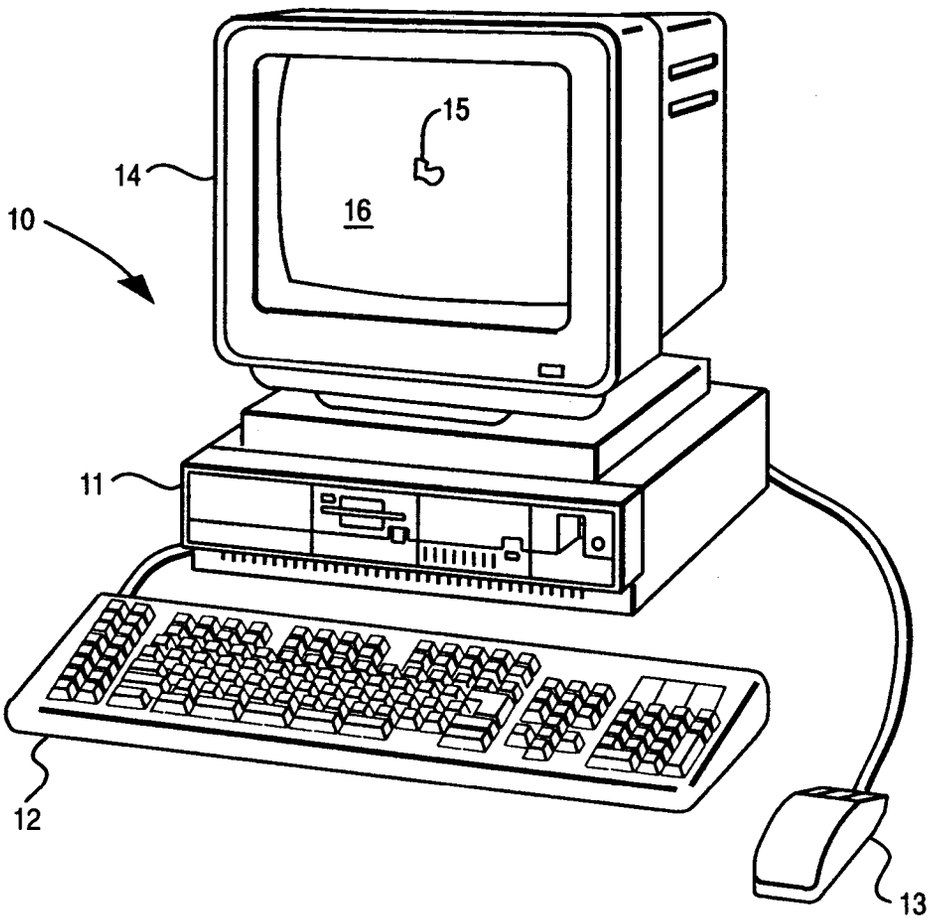


FIG. 1

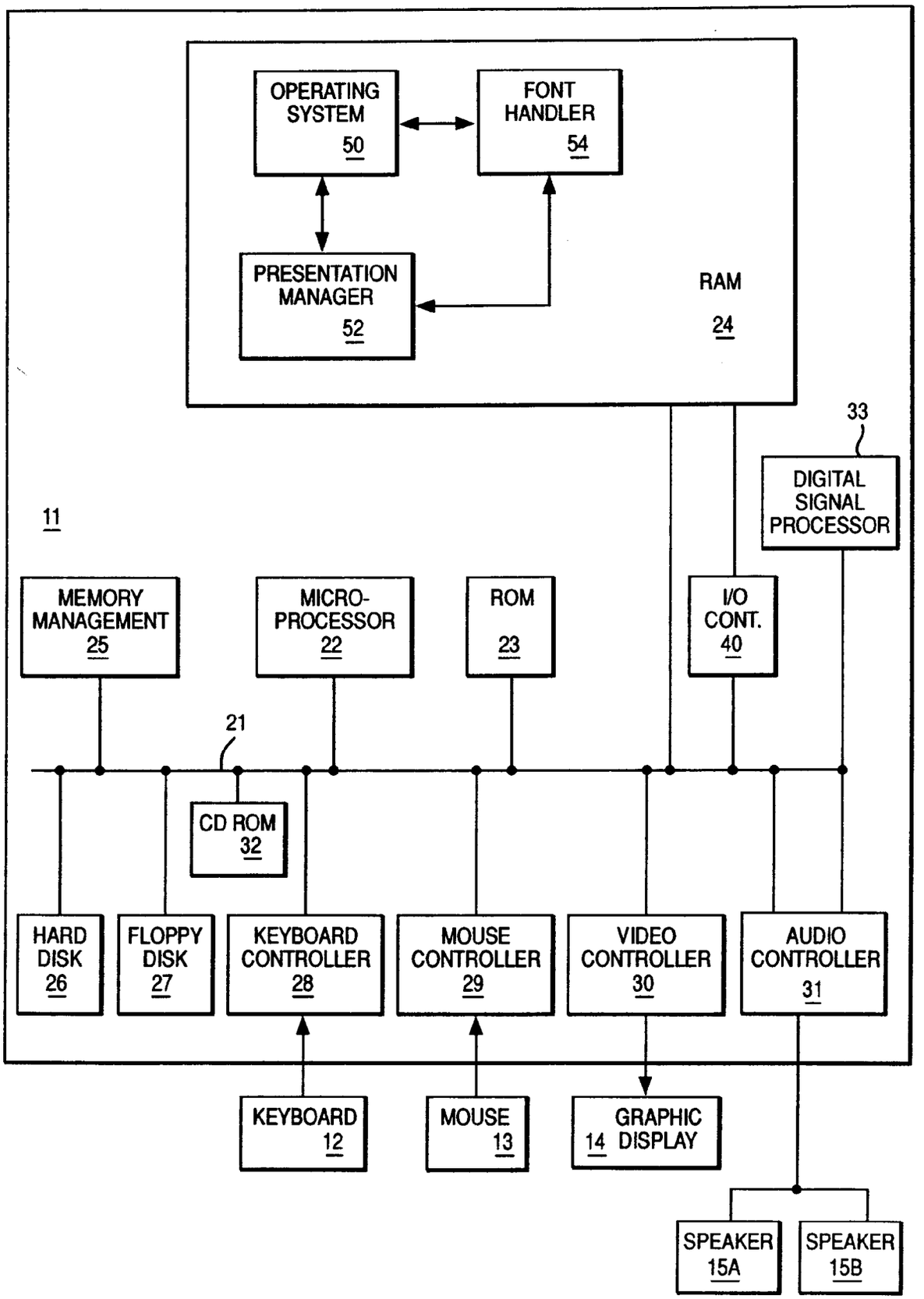
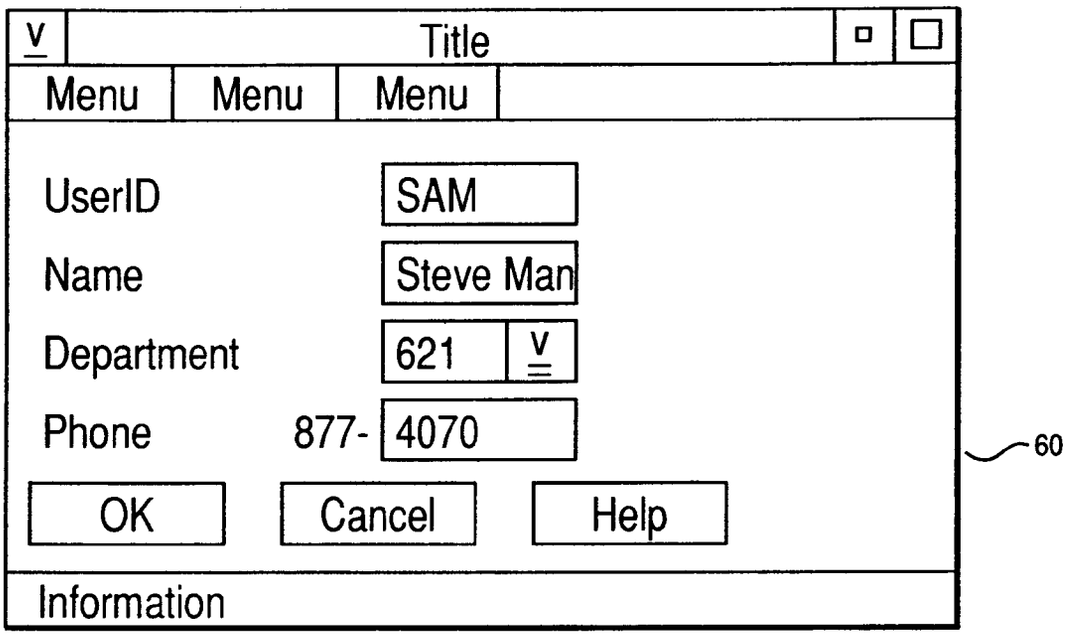
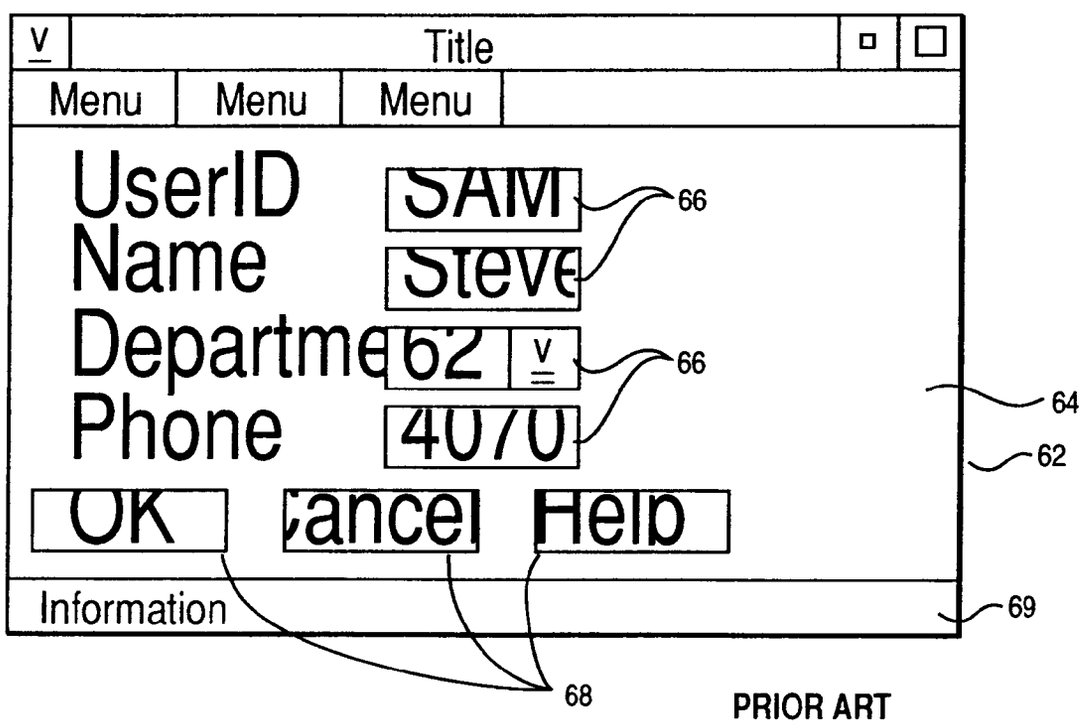

OPERATING SYSTEM 50
FONT HANDLER 54
PRESENTATION MANAGER 52
RAM 24
33
DIGITAL SIGNAL PROCESSOR
11
MEMORY MANAGEMENT 25
MICRO-PROCESSOR 22
ROM 23
I/O CONT. 40
21
CD ROM 32
HARD DISK 26
FLOPPY DISK 27
KEYBOARD CONTROLLER 28
MOUSE CONTROLLER 29
VIDEO CONTROLLER 30
AUDIO CONTROLLER 31
KEYBOARD 12
MOUSE 13
GRAPHIC 14 DISPLAY
SPEAKER 15A
SPEAKER 15B

FIG. 2



PRIOR ART

FIG. 3A



PRIOR ART

FIG. 3B

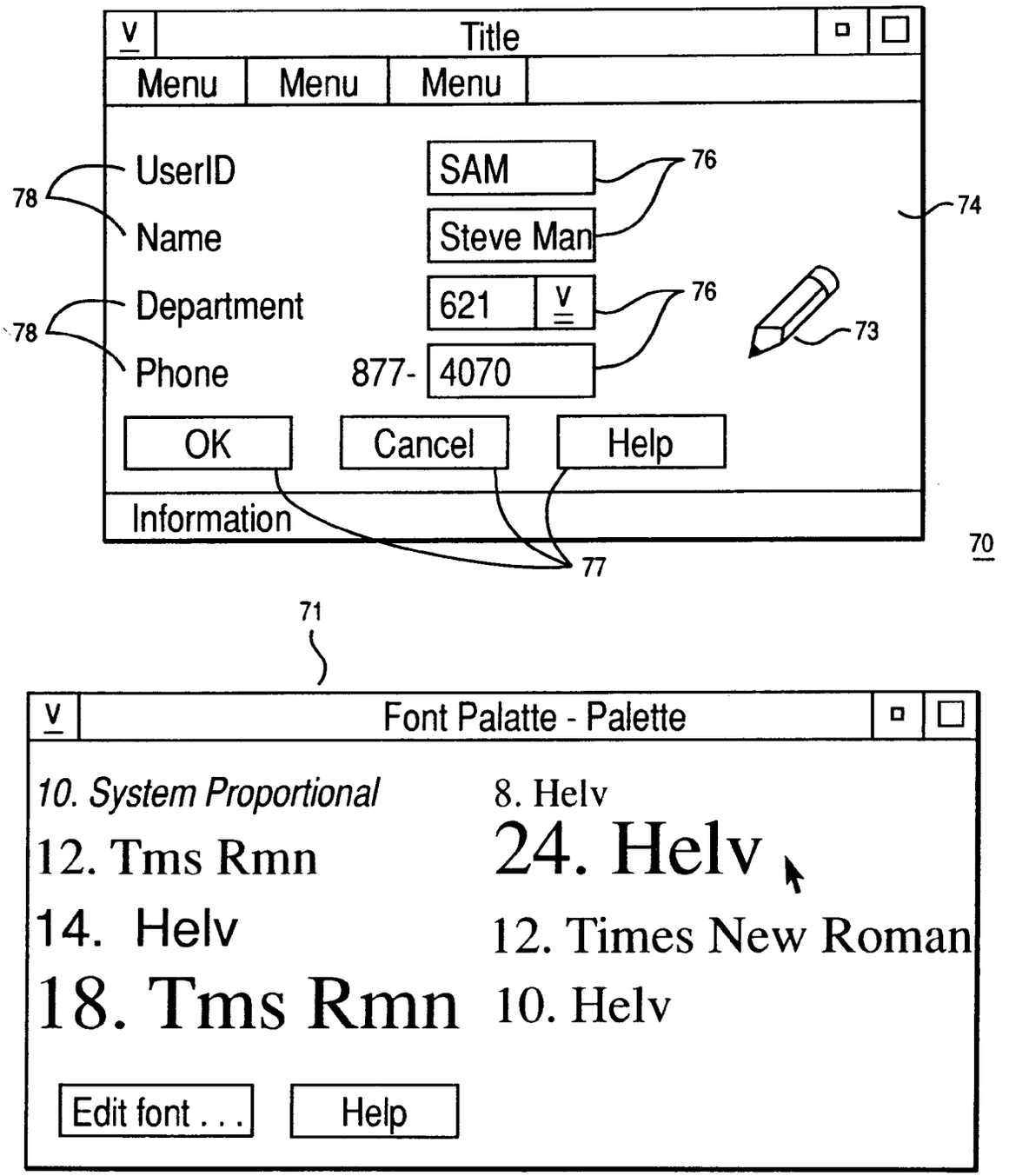

Title
Menu
Menu
Menu
UserID
SAM
Name
Steve Man
Department
621
Phone
877-
4070
OK
Cancel
Help
Information
78
78
76
76
74
73
77
70
71
Font Palatte - Palette
10. System Proportional
12. Tms Rmn
14. Helv
18. Tms Rmn
8. Helv
24. Helv
12. Times New Roman
10. Helv
Edit font . . .
Help

FIG. 4A

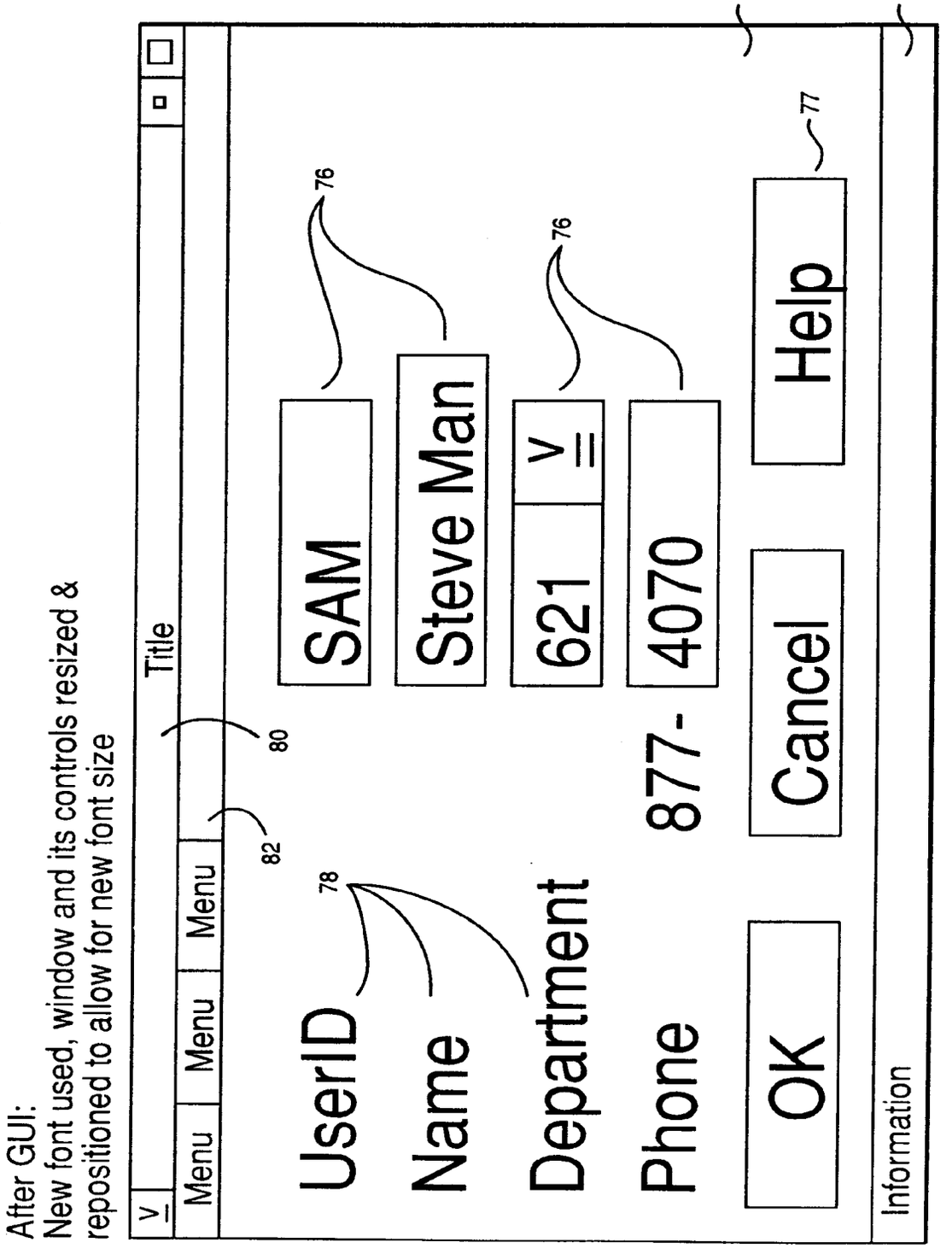

After GUI:
New font used, window and its controls resized & repositioned to allow for new font size
Title
Menu
Menu
Menu
UserID
Name
Department
Phone
SAM
Steve Man
621
877- 4070
OK
Cancel
Help
Information
78
76
76
77
74
70
80
82

FIG. 4B

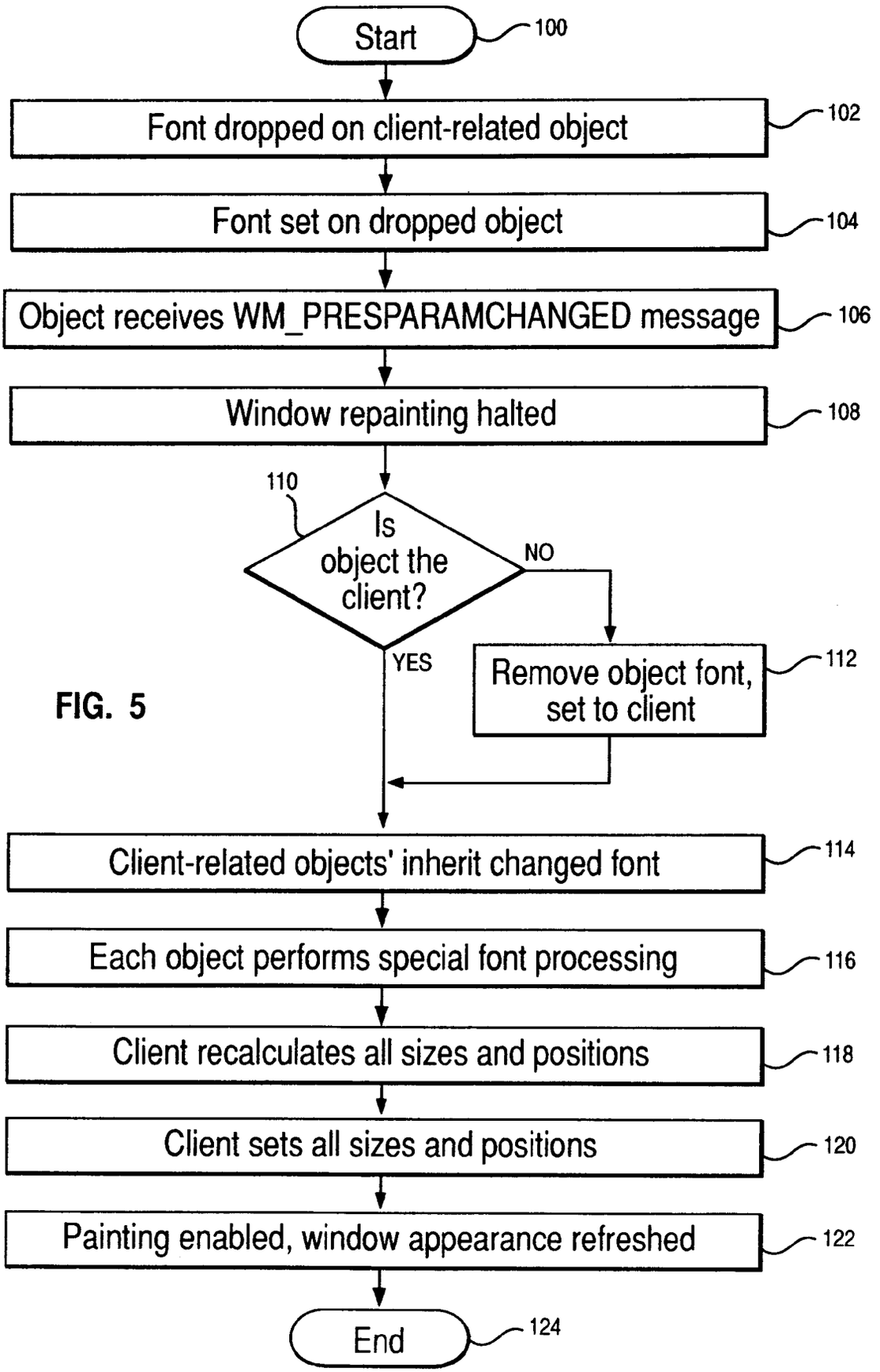

Start
100
Font dropped on client-related object
102
Font set on dropped object
104
Object receives WM_PRESPARAMCHANGED message
106
Window repainting halted
108
110
Is object the client?
NO
YES
Remove object font, set to client
112
Client-related objects' inherit changed font
114
Each object performs special font processing
116
Client recalculates all sizes and positions
118
Client sets all sizes and positions
120
Painting enabled, window appearance refreshed
122
End
124

FIG. 5

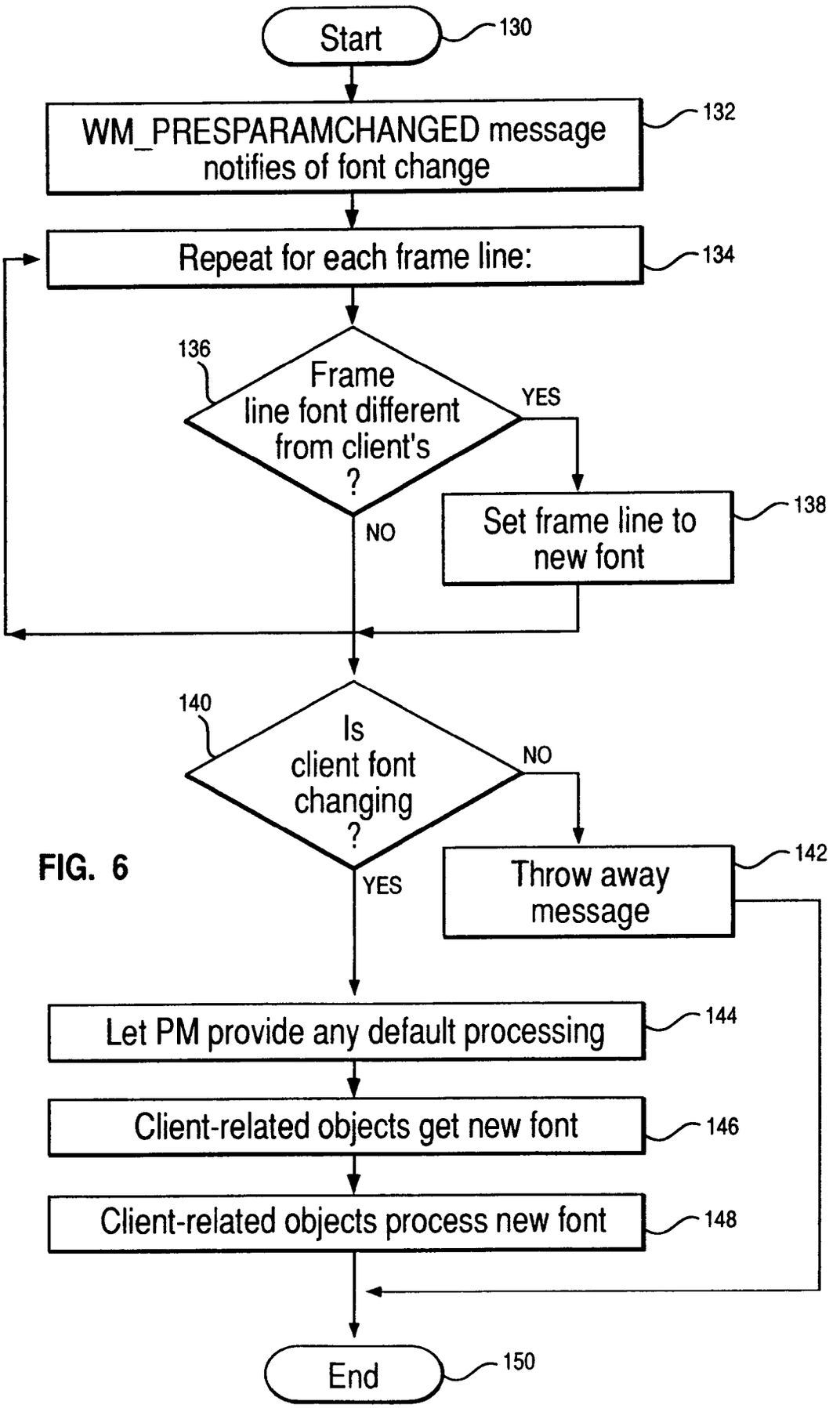

Start
130
WM_PRESPARAMCHANGED message notifies of font change
132
Repeat for each frame line:
134
136
Frame line font different from client's ?
YES
NO
Set frame line to new font
138
140
Is client font changing ?
NO
YES
Throw away message
142
Let PM provide any default processing
144
Client-related objects get new font
146
Client-related objects process new font
148
End
150

FIG. 6

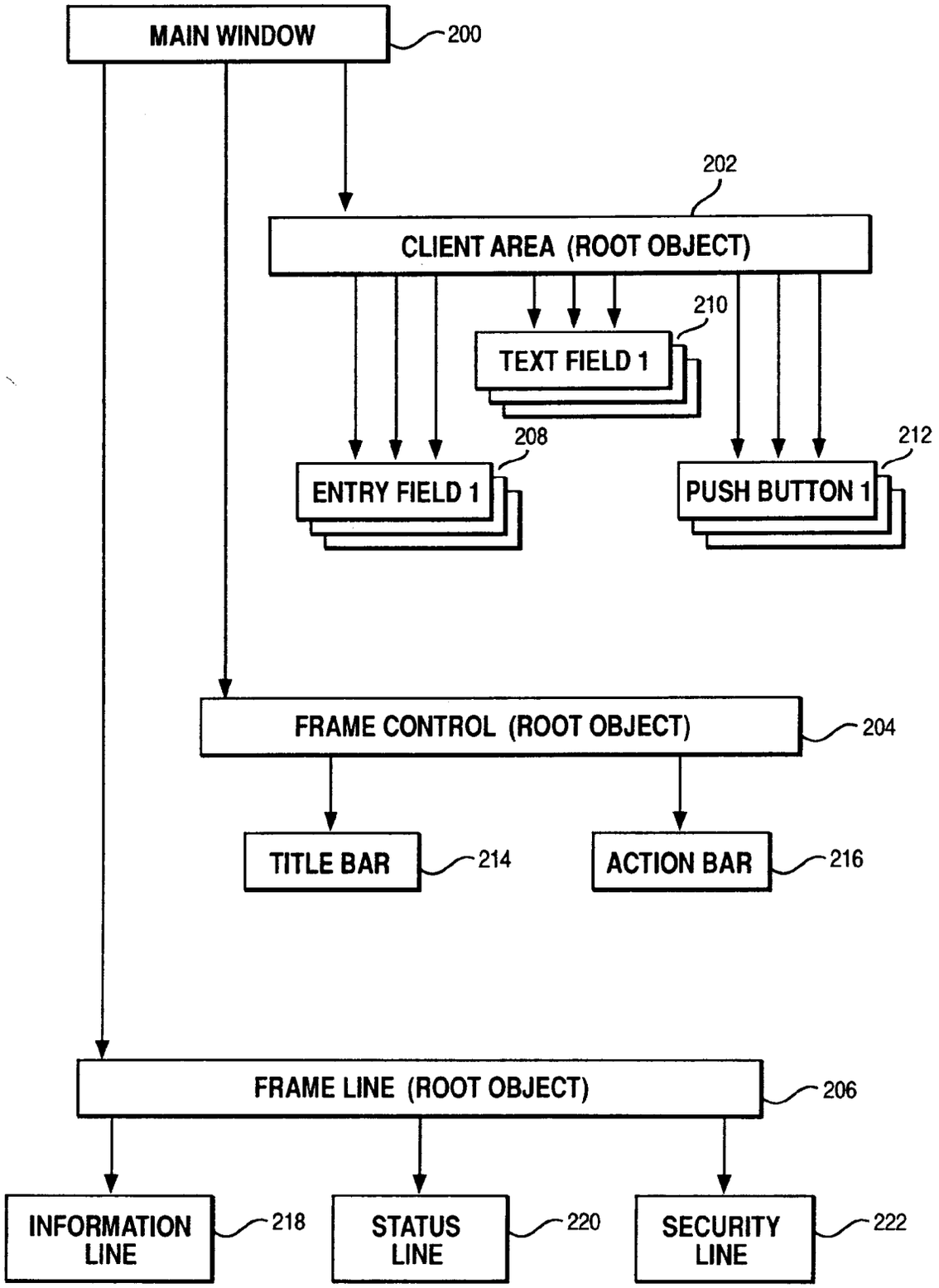

MAIN WINDOW
200
202
CLIENT AREA (ROOT OBJECT)
210
TEXT FIELD 1
208
ENTRY FIELD 1
212
PUSH BUTTON 1
FRAME CONTROL (ROOT OBJECT)
204
TITLE BAR
214
ACTION BAR
216
FRAME LINE (ROOT OBJECT)
206
INFORMATION LINE
218
STATUS LINE
220
SECURITY LINE
222

FIG. 7

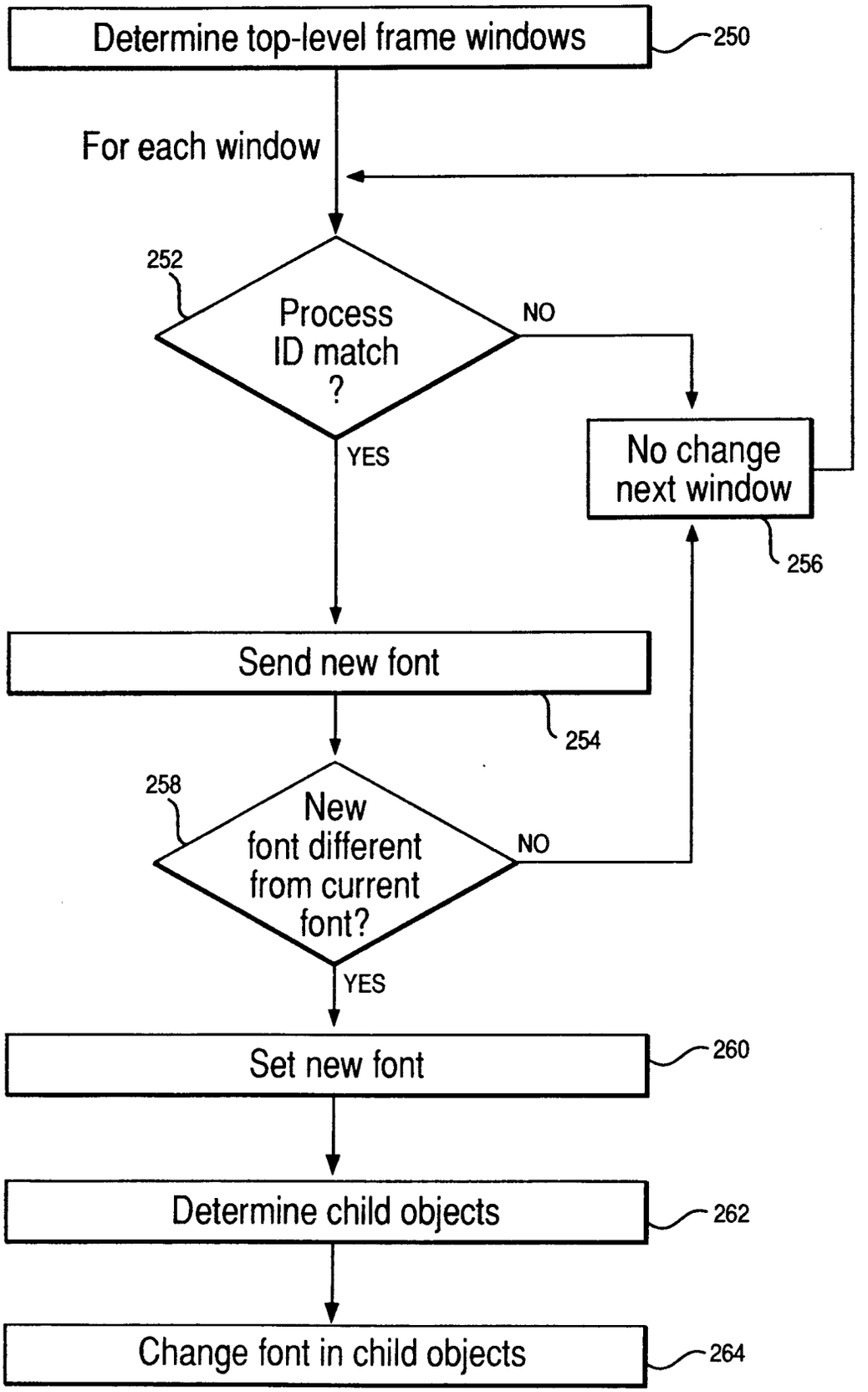

Determine top-level frame windows
250
For each window
252
Process ID match ?
NO
YES
No change next window
256
Send new font
254
258
New font different from current font?
NO
YES
Set new font
260
Determine child objects
262
Change font in child objects
264

FIG. 8

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 94 30 2629

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.5)
X	IBM TECHNICAL DISCLOSURE BULLETIN vol. 35, no. 7 , December 1992 , NEW YORK US pages 285 - 286 'SELECTING OF A USABLE FONT SIZE FOR THE HOST SESSIONS' * page 285, line 1 - line 26 *	1,8	G06F3/033
Y		2,3,5-7, 9,10, 12-14	
Y	IBM TECHNICAL DISCLOSURE BULLETIN vol. 27, no. 12 , May 1985 , NEW YORK US page 7066 'UPDATING ATTRIBUTES OR DATA OF PARENT VIA CHANGES TO THE CHILD' * page 7066, line 1 - line 16 *	2,3,5,6, 9,10,12, 13	
Y	LEVENSON S. ET AL 'Now That I Have OS/2 2.0 On My Computer What Do I Do Next ...' 1992 , VAN NOSTRAND REINHOLD , NEW YORK * page 90, line 1 - line 9 *	7,14	
A		4,11	G06F

TECHNICAL FIELDS SEARCHED (Int.Cl.5)

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	29 July 1994	Bailas, A

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C08)